Individual Project Assignment

CE4024-CZ402 CRYPTOGRAPHY & NETWORK SECURITY

If you are subscribed to CPE413 or CSC409, an additional project will have to be done in addition to the one described here. Please identify yourself to the lecturers immediately.

SUBMISSION DEADLINE: MONDAY, 17 APRIL 2017 23:30

Objectives:

The main objective of this assignment is for the students to learn the applications of cryptographic primitives. In particular:

- To understand how the application of encryption mode for block cipher can affect the security of the encrypted text.
- To evaluate a (flaw) design of message authentication code for security vulnerability.

Important notes:

The project needs to be done by students individually. Any plagiarism, in the implementation code or the report, if identified, will lead to forfeiture of all marks for the project, and further disciplinary actions as per the university rules may be taken. For the same reason, students are advised not to share their work with other students or in any public forum/private forums, since we will not distinguish the creators of specific content from the copiers.

The **deadline for submitting the solutions is Monday, 17 APR 2017, 23:30**. **Delay by calendar day will entail 20% penalty per day.** So, for example, if a student submits the solutions on anytime on the calendar day of 20 Apr 2017, then s/he will receive only 40% of whatever would be the score s/he would have received for the technical merit, had it been submitted before the deadline.

The assignment consists of two problems. For each problem, you are asked to write a Java program to implement the solution. You are also asked to write a report explaining your solution. Part of the grading will be done by executing the student's solutions with automated scripts. Consequently, adherence to the submission guidelines (see below), including the filenames need to be followed strictly. **Failing to do so will lead to a penalty of 25% per affected problem**.

We have created project templates for students to use to solve the problems. The provided templates include the necessary cryptographic library to solve the problem. **The students are not allowed to use any other libraries other than those provided in the template**. **Failure to comply with this requirement may result in the submissions to fail our automated tests, in which case the student will have a penalty of 100%**.

The provided templates have been created by Phd student Mr. Chen Hongxu with the lecturers' guidance, and Mr. Chen will be the person who will also do the actual evaluation of the projects,

albeit under the supervision of the lecturers. So if there are any technical issues, please contact him directly at **HCHEN017@e.ntu.edu.sg** but please put at least one of the lecturers in cc, so that we are in the loop, and are aware of issues and can act in a timely manner if any need arises. A rubric for the technical assessment is provided next.

Assessment rubric

There are two problems in this assignment. For each problem, the student is asked to write a Java program to implement the solution to the problem, and to write a report explaining the solution (how and why it works). The marks are divided equally among the two problems. For each problem, the breakdown of the marks is as follows:

- Correct implementation of code: 70%.
- Report: 30%.

The code will be primarily tested using an automated tool. The report needs to be concise and explain only the vulnerabilities you are exploiting and the design of your attack. There is no need to explain in detail every line of code, but only the essence of the algorithm you use in your attack.

Note: Information about the submission process (through NTUlearn) will be provided separately, closer to the submission date.

Problem Description

Backgrounds

In cryptography, to analyze the strength of certain cryptographic functions, it is customary to consider a hypothetical `oracle' which may perform certain tasks when queried. Such an oracle is assumed to have in its possession certain secrets, such as the encryption/decryption key, and the task of the attacker can be to figure one or more secrets, or forge certain messages (MAC or digital signatures) by interacting with the oracle. For example, an 'encryption oracle' may produce a ciphertext that corresponds to the plain text supplied by the attacker, but would not disclose the secret key it uses to encrypt the plain text. In such a scenario, one could ask whether, given enough plaintext-ciphertxt pairs, the attacker could figure out the secret key or not. The following two problems will assume certain oracles that can perform certain tasks when queried, and the goal of the attacker is to guess certain secrets possessed by the oracle (Problem 1), or forge a message authentication code that can pass the test by the oracle (Problem 2). Implementation of the Oracle will be provided to the students.

Problem 1

We assume an encryption oracle that possesses a secret key k and a 64-bit block of secret value s that are known only to the oracle initially. We assume all encryption is done using DES with 56-bit key and input block size of 64 bit, and that encryption is done in ECB mode.

The encryption oracle, when given a message m, will respond by outputting the message $E(k, m')$ where m' is obtained as follows:

- If $m \parallel s$ is a multiple of 64 bits, then $m' = m \parallel s$. (note that is ‖ denotes the append operation)
- Otherwise, m' is obtained by adding the minimum amount of trailing 0's to $m \parallel s$ until its size is a multiple of 64 bits.

Assume that the attacker can query the oracle as many times as needed. Your task is to implement an attack to determine the secret s from the oracle. The attack must work without the attacker knowing the encryption key k. The attack code must succeed within a time limit of 2 minutes.[1]

Deliverable:

Implement your attack in the class template `Attack1` in the provided project template. Put all your codes in the class `Attack1`. Read the README.txt file provided in the template for further information about the project template.

You may change the values of the parameters of the oracle to help with your testing and debugging, but do not modify any other parts of the project template. You must only submit the following two files for this problem:

- The source file **Attack1.java**. DO NOT submit any other source files. Any other source or binary files other than Attack1.java will be discarded during the assessment. So make sure you do not use any external libraries other than those provided in the template.
- A PDF file **Attack1.pdf**, explaining the vulnerability you have identified, and the design of your attack, in not more than 2 pages of A4-size paper, and using size 12 Times New Roman font (or other fonts of equivalent size).

Important Note:

The provided template comes with fixed values for key k and secret s. These will be changed during the assessment so your solution must not depend on the knowledge of these values.

Problem 2

In this problem we assume a given oracle that performs calculations related to message authentication code (MAC).

The oracle implements the following MAC algorithm: Given a MAC key k and an input message $d = d_1 \parallel \cdots \parallel d_n$ consisting of n 64-bit blocks of bits, we first calculate blocks $o_1, \ldots, o_n$ as follows:

[1] This is an estimate assuming it runs in typical recent laptop computer (say, i5 4th gen CPU, 8GB RAM). Note that this is a very conservative estimate and the actual attack, if implemented correctly, will take only seconds at most.

$$\begin{aligned} o_1 &= E(k, d_1) \\ o_2 &= E(k, o_1 \oplus d_2) \\ \vdots & \quad \vdots \\ o_n &= E(k, o_{n-1} \oplus d_n) \end{aligned}$$

where E is the DES encryption function and $\oplus$ is the bitwise exclusive-or operator. The MAC value of d is calculated as follows:

$$MAC(k, d) = o_1 \oplus \cdots \oplus o_n.$$

Note that in the case where n = 1, we have $MAC(k, d) = E(k, d_1)$.

The oracle keeps the MAC key k which is known only to the oracle. The oracle provides two functions that the attacker can invoke:

- $mac0(n)$: this function takes an input n which is a positive integer, and returns $MAC(k, d)$ where d is a sequence of zeros of length n blocks.
- $mac3(m)$: this function takes an input m which is a message of length 3 blocks and outputs $MAC(k, m)$. If m is longer than three blocks, the function truncates the message to 3 block length. If m is less than three-blocks long, trailing zeros are added to m before the MAC is returned.

Furthermore, following function is provided to check the correctness of a given MAC value for a given message:

- $check(m, c)$: this function takes an input message m of arbitrary size and a one-block input c. Let m' be the message obtained from m by adding the minimum number of trailing zeros so that the length of m' is a multiple of block size. It returns `true' if $c = MAC(k, m')$. Otherwise it returns false.

Your task is to implement a universal forgery attack. That is, for any given message m of arbitrary length, compute the value $MAC(k, m)$ without knowing the key k. The attacker can use the oracle functions given above as many times as needed. The timing constraint for the attack to succeed is the same as in Problem 1.

Deliverable:

Implement your attack in the class template `Attack2` in the provided project template. Put all your codes in the class `Attack2`. Read the README.txt file provided in the template for further information about the project template.

You may change the values of the parameters of the oracle to help with your testing and debugging, but do not modify any other parts of the project template. You must only submit the following two files for this problem:

- The source file **Attack2.java**. DO NOT submit any other source files. Any other source or binary files other than Attack2.java will be discarded during the assessment. So make sure you do not use any external libraries other than those provided in the template.

- A PDF file **Attack2.pdf**, containing your answers for the questions in the theory part, not more than 2 pages of A4-size paper, and using size 12 Times New Roman font (or other fonts of equivalent size).

What to submit

Put all the deliverables (Attack1.java, Attack1.pdf, Attack2.java, Attack2.pdf) in one .zip archive and name it as follows: [Matr. No.]_[Full name].zip. For example, if your matriculation number is U123456 and your name is James Ong, then name your file as U123456_JAMES_ONG.zip (spaces in name replaced by underscore). Note that if the submission does not adhere strictly to this specification, you may risk receiving zero marks for the assessment.

Waiver of penalties:

Any waiver of any of the aforementioned penalties is solely up to the discretion of the lecturers, on a case by case basis.